Implement a C program to eliminate left recursion from a given CFG.

```
#include <stdio.h>
#include <string.h>

#define MAX_RULES 10
#define MAX_LEN 50

char grammar[MAX_RULES][MAX_LEN];  // Stores the grammar rules
int ruleCount;                     // Number of rules

// Function to remove left recursion
void removeLeftRecursion() {
   printf("\nGrammar after eliminating left recursion:\n");

   for (int i = 0; i < ruleCount; i++) {
      char *rule = grammar[i];
      char nonTerminal = rule[0];  // Left-hand side (LHS)
      char alpha[MAX_LEN] = "", beta[MAX_LEN] = "";
      int hasLeftRecursion = 0;

      // Parsing right-hand side (RHS)
      char *rhs = strtok(&rule[3], "|"); // Skip "A->"
      while (rhs != NULL) {
         if (rhs[0] == nonTerminal) {
            hasLeftRecursion = 1;
            strcat(alpha, rhs + 1); // Store recursive part (skip `A`)
            strcat(alpha, "|");
         } else {
            strcat(beta, rhs); // Store non-recursive part
            strcat(beta, "|");
         }
```

```
            rhs = strtok(NULL, "|");
        }

        if (hasLeftRecursion) {
            // Remove trailing '|'
            if (beta[strlen(beta) - 1] == '|') beta[strlen(beta) - 1] = '\0';
            if (alpha[strlen(alpha) - 1] == '|') alpha[strlen(alpha) - 1] = '\0';

            printf("%c -> %s%c'\n", nonTerminal, beta, nonTerminal);
            printf("%c' -> %s%c' | ε\n", nonTerminal, alpha, nonTerminal);
        } else {
            printf("%s\n", grammar[i]); // No change if no left recursion
        }
    }
}

// Main function
int main() {
    printf("Enter the number of grammar rules: ");
    scanf("%d", &ruleCount);
    getchar(); // Consume newline

    printf("Enter the grammar rules (format: A->Aα|β, use 'ε' for epsilon):\n");
    for (int i = 0; i < ruleCount; i++) {
        fgets(grammar[i], MAX_LEN, stdin);
        grammar[i][strcspn(grammar[i], "\n")] = '\0';  // Remove newline
    }

    removeLeftRecursion();
    return 0;
}
```

Input:

Enter the number of grammar rules: 2

Enter the grammar rules (format: A->Aα|β, use 'ε' for epsilon):

A->Aα|β

B->Bγ|δ

Output: